Practitioner's Docket No. NAI1P467/00.177.01 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: Neil Andrew Cowie et al.

Application No.: 09/912,389 Group No.: 2131
Filed: July 26, 2001 Examiner: Henning, Matthew T.
For: DETECTING COMPUTER PROGRAMS WITHIN PACKED COMPUTER FILES

Mail Stop Appeal Briefs – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF
(PATENT APPLICATION--37 C.F.R. § 41.37)

1. This brief is in furtherance of the Notice of Appeal, filed in this case on 12/17/2007, and in response to the Notice of Panel Decision from Pre-Appeal Brief Review, mailed 02/04/2008.

2. STATUS OF APPLICANT

   This application is on behalf of other than a small entity.

3. FEE FOR FILING APPEAL BRIEF

   Pursuant to 37 C.F.R. § 41.20(b)(2), the fee for filing the Appeal Brief is:

other than a small entity	$510.00
Appeal Brief fee due	**$510.00**

4. EXTENSION OF TERM

   The proceedings herein are for a patent application and the provisions of 37 C.F.R. § 1.136 apply.

   Applicant believes that no extension of term is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

5. TOTAL FEE DUE

   The total fee due is:

Appeal brief fee	$510.00
Extension fee (if any)	$0.00
TOTAL FEE DUE	**$510.00**

6. FEE PAYMENT

Authorization is hereby made to charge the amount of $510.00 to Deposit Account No. 50-1351 (Order No. NAI1P467).

7. FEE DEFICIENCY

If any additional extension and/or fee is required, and if any additional fee for claims is required, charge Deposit Account No. 50-1351 (Order No. NAI1P467).

Date: March 4, 2008

/KEVINZILKA/
Signature of Practitioner

Reg. No.: 41,429
Tel. No.: 408-971-2573
Customer No.: 28875

Kevin J. Zilka
Zilka-Kotab, PC
P.O. Box 721120
San Jose, CA 95172-1120

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	)
	)
Cowie et al.	) Art Unit: 2131
	)
Application No. 09/912,389	) Examiner: Henning, Matthew T.
	)
Filed: 07/26/2001	) Atty. Docket No.:
	) NAI1P467/00.177.01
For: DETECTING COMPUTER PROGRAMS	)
WITHIN PACKED COMPUTER FILES	) Date: 03/04/2008
	)
	)

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

APPEAL BRIEF (37 C.F.R. § 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on 12/17/2007, and in response to the Notice of Panel Decision from Pre-Appeal Brief Review, mailed 02/04/2008.

The fees required under § 1.17, and any required petition for extension of time for filing this brief and fees therefor, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief contains these items under the following headings, and in the order set forth below (37 C.F.R. § 41.37(c)(i)):

I REAL PARTY IN INTEREST
II RELATED APPEALS AND INTERFERENCES
III STATUS OF CLAIMS
IV STATUS OF AMENDMENTS
V SUMMARY OF CLAIMED SUBJECT MATTER

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

VII ARGUMENT

VIII CLAIMS APPENDIX

IX EVIDENCE APPENDIX

X RELATED PROCEEDING APPENDIX

The final page of this brief bears the practitioner's signature.

I REAL PARTY IN INTEREST (37 C.F.R. § 41.37(c)(1)(i))

The real party in interest in this appeal is McAfee, Inc.

II RELATED APPEALS AND INTERFERENCES (37 C.F.R. § 41.37(c) (1)(ii))

With respect to other prior or pending appeals, interferences, or related judicial proceedings that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no other such appeals, interferences, or related judicial proceedings.

A Related Proceedings Appendix is appended hereto.

III STATUS OF CLAIMS (37 C.F.R. § 41.37(c) (1)(iii))

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98.

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims withdrawn from consideration: None
2. Claims pending: 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98.
3. Claims allowed: None
4. Claims rejected: 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98.
5. Claims cancelled: 4, 9-11, 13, 20, 25-27, 29, 36, 41-43, 45, 52, 57-59, 61, 68, 73-75, 77, 84, 89-91, 93 and 97.

C. CLAIMS ON APPEAL

The claims on appeal are: 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98.

See additional status information in the Appendix of Claims.

IV STATUS OF AMENDMENTS (37 C.F.R. § 41.37(c)(1)(iv))

As to the status of any amendment filed subsequent to final rejection, the Amendments submitted on 10/12/2005 and 11/27/2006 were not entered by the Examiner.

V SUMMARY OF CLAIMED SUBJECT MATTER (37 C.F.R. § 41.37(c)(1)(v))

With respect to a summary of Claim 1, as shown in Figures 1, 2, 3, 4a and 6 et al., a computer program product in a computer storage medium comprises a computer program operable to control a computer to detect a known computer program within a packed computer file (e.g. see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution. Further, the computer program product comprises resource data reading logic for reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Additionally, the computer program product comprises resource data comparing logic for generating characteristics of the resource data and for comparing the characteristics of the resource data with characteristics of known computer program resource data (e.g. see item 42 of Figure 6, etc.) and for detecting a match (e.g. see item 44 of Figure 6, etc.) with the known computer program indicative of the packed computer file containing the known computer program.

Furthermore, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. In addition, the generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. Moreover, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Further, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Additionally, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of the program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with the program resource items within the resource data of the packed computer file, and sizes of the program resource items within the resource data of the packed computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17,

and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; page 6, lines 25-26; and page 11, lines 22-25 et al.

With respect to a summary of Claim 17, as shown in Figures 1, 2, 3, 4a and 6 et al., a computer program product in a computer storage medium comprises a computer program operable to control a computer to generate data for detecting a known computer program within a packed computer file (e.g. see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution. Further, the computer program product comprises resource data reading logic for reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Additionally, the computer program product comprises characteristic data generating logic for generating characteristic data associated with the resource data for comparison with characteristic data of known computer program resource data (e.g. see item 42 of Figure 6, etc.) to detect a match (e.g. see item 44 of Figure 6, etc.) with the known computer program indicative of the packed computer file containing the known computer program.

Furthermore, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. In addition, generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. Moreover, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Further, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Additionally, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with program resource items within the resource data of the packed computer file, and sizes of program resource items within the resource data of the packed

computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17, and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; page 6, lines 25-26; and page 11, lines 22-25 et al.

With respect to a summary of Claim 33, as shown in Figures 1, 2, 3, 4a and 6 et al., a method of controlling a computer to detect a known computer program within a packed computer file (e.g. see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution, comprises the steps of reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Further, the method comprises the steps of generating characteristics of the resource data and comparing the characteristics of the resource data with characteristics of known computer program resource data (e.g. see item 42 of Figure 6, etc.) and detecting a match (e.g. see item 44 of Figure 6, etc.) with characteristics of the known computer program indicative of the packed computer file containing the known computer program.

Additionally, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. Furthermore, the generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. In addition, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Moreover, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Further, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with program resource items within the resource data of the packed computer file, and sizes of program resource items within the resource data of the packed

computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17, and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; and page 6, lines 25-26 et al.

With respect to a summary of Claim 49, as shown in Figures 1, 2, 3, 4a and 6 et al., a method of controlling a computer to generate data for detecting a known computer program within a packed computer file (e.g. see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution, comprises the steps of reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Further, the method comprises the steps of generating characteristic data associated with the resource data for comparison with characteristic data of known computer program resource data (e.g. see item 42 of Figure 6, etc.) and detecting a match (e.g. see item 44 of Figure 6, etc.) with the known computer program indicative of the packed computer file containing the known computer program.

Additionally, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. Furthermore, the generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. In addition, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Moreover, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Further, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with program resource items within the resource data of the packed computer file, and sizes of program resource items within the resource data of the packed computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17, and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; and page 6, lines 25-26 et al.

With respect to a summary of Claim 65, as shown in Figures 1, 2, 3, 4a and 6 et al., an apparatus for detecting a known computer program within a packed computer file (e.g. see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution, comprises a resource data reader for reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Further, the apparatus comprises a resource data comparator for generating characteristics of the resource data and for comparing the characteristics of the resource data with characteristics of known computer program resource data (e.g. see item 42 of Figure 6, etc.) for detecting a match (e.g. see item 44 of Figure 6, etc.) with the known computer program indicative of the packed computer file containing the known computer program.

Additionally, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. Furthermore, the generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. In addition, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Moreover, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Further, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with program resource items within the resource data of the packed computer file, and sizes of program resource items within the resource data of the packed computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17, and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; and page 6, lines 25-26 et al.

With respect to a summary of Claim 81, as shown in Figures 1, 2, 3, 4a and 6 et al., an apparatus for generating data for detecting a known computer program within a packed computer file (e.g.

see item 5 of Figure 1, etc.), with the packed computer file being unpacked upon execution, comprises a resource data reader for reading resource data (e.g. see item 6 of Figure 1, etc.) within the packed computer file, where the resource data specifies program resource items used by the known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by the packed computer file. Further, the apparatus comprises a characteristic data generator for generating characteristic data associated with the resource data for comparison with characteristic data of known computer program resource data (e.g. see item 42 of Figure 6, etc.) and for detecting a match (e.g. see item 44 of Figure 6, etc.) with the known computer program indicative of the packed computer file containing the known computer program.

Additionally, the resource data of the packed computer file is processed to generate fingerprint data (e.g. see Figure 4a, etc.) and to compare the generated fingerprint data with known computer program fingerprint data. Furthermore, the generated fingerprint data includes a number of program resource items specified within the resource data of the packed computer file. In addition, the generated fingerprint data includes a flag indicating which data is included within the generated fingerprint data. Moreover, the generated fingerprint data includes a location within the resource data of the packed computer file of an entry specifying a program resource item having a largest size.

Further, the generated fingerprint data includes a checksum value (e.g. see Figure 3, etc.) calculated in dependence upon a number of program resource items specified beneath each node within hierarchically arranged resource data (e.g. see Figure 2, etc.) of the packed computer file, string names associated with program resource items within the resource data of the packed computer file, and sizes of program resource items within the resource data of the packed computer file. See, for example, page 2, line 26 – page 3, line 6; page 4, lines 1-4, 14-17, and 22-24; page 4, line 30 – page 5, line 3; page 5, lines 11-12; and page 6, lines 25-26 et al.

Of course, the above citations are merely examples of the above claim language and should not be construed as limiting in any manner.

VI GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL (37 C.F.R. § 41.37(c)(1)(vi))

Following, under each issue listed, is a concise statement setting forth the corresponding ground of rejection.

Issue # 1: The Examiner has rejected Claims 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98 under 35 U.S.C. 103(a) as being unpatentable over Cozza (U.S. Patent No. 5,649,095), in view of Arnold et al. (U.S. Patent No. 5,442,699), and further in view of Pietrek ("Peering Inside the PE: A Tour of the Win 32 Portable Executable").

VII ARGUMENT (37 C.F.R. § 41.37(c)(1)(vii))

The claims of the groups noted below do not stand or fall together. In the present section, appellant explains why the claims of each group are believed to be separately patentable.

Issue # 1:

The Examiner has rejected Claims 1-3, 5-8, 12, 14-19, 21-24, 28, 30-35, 37-40, 44, 46-51, 53-56, 60, 62-67, 69-72, 76, 78-83, 85-88, 92, 94-96, and 98 under 35 U.S.C. 103(a) as being unpatentable over Cozza (U.S. Patent No. 5,649,095), in view of Arnold et al. (U.S. Patent No. 5,442,699), and further in view of Pietrek ("Peering Inside the PE: A Tour of the Win 32 Portable Executable").

Group #1: Claims 1-3, 5-8, 12, 16-19, 21-24, 28, 32-35, 37-40, 44, 48-51, 53-56, 60, 64-67, 69-72, 76, 80-83, 85-88, 92, and 96

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on appellant's disclosure. In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

With respect to the first element of the *prima facie* case of obviousness and, in particular, the obviousness of combining the aforementioned references, the Examiner has argued that "it ... would have been obvious to the ordinary person skilled in the art at the time of [the] invention to employ the teachings of Arnold in the virus scanning of Cozza by creating hashes of the data, including the resources, of the compressed file and comparing it to hashes of known viral patterns." To the contrary, appellant respectfully asserts that it would not have been obvious to

combine the teachings of the Cozza and Arnold references, especially in view of the vast evidence to the contrary.

The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. In re Mills, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990). Although a prior art device "may be capable of being modified to run the way the apparatus is claimed, there must be a suggestion or motivation in the reference to do so." 916 F.2d at 682, 16 USPQ2d at 1432.).

Appellant respectfully points out that the Cozza reference teaches that "[i]f... [there is a] need to scan for one or more viruses then a check is made to see whether the file is compressed" (Col. 7, lines 16-19). Cozza further discloses that "[a] compressed file may have already been decompressed" but that "[i]f the file does require decompression, then it is decompressed" (Col. 7, lines 19-21). In fact, Cozza only teaches that after such decompression, the file is scanned (see steps 84, 86, 90 and 94 in Figure 4D of Cozza).

However, the Arnold reference merely relates to "automatic[ally] search[ing] for one or more patterns which may be contained within a body of text or computer data in encrypted form and, more particularly, to searching for encrypted patterns within computer viruses" (Col. 1, lines 18-22 – emphasis added). In fact, appellant respectfully points out that Arnold expressly discloses "detect[ing] and locat[ing] patterns that are present within data that has been encrypted" (see Abstract), where "for those patterns where it is of interest to detect, in the text 44, encryptions of the pattern rather than just the pattern itself (e.g. patterns from self-encrypting computer viruses), the signatures may be subjected to an invariant transformation appropriate to the type of encryption" (Col. 4, lines 38-44). Clearly, Arnold only teaches searching for patterns within encrypted data. To this end, there would have been no motivation or suggestion to combine Arnold, which only relates to searching for patterns within encrypted data, with the teachings of Cozza, which only scans data after such data is decompressed. Thus, the Examiner's proposed combination is inappropriate.

Further, the fact that Cozza teaches that decompression takes place before scanning for viruses actually *teaches away* from "creating hashes of data... of the compressed file" in order to "scan

the files as quickly as possible" (emphasis added), as alleged by the Examiner to be taught in Arnold. It is improper to combine references where the references teach away from their combination. *In re Grasselli*, 713 F.2d 731, 743, 218 USPQ 769, 779 (Fed. Cir. 1983).

In the Office Action mailed 09/17/2007, the Examiner has argued that "Cozza discloses that if the file requires decompression, then it is decompressed," and that "[t]his does not require decompression in Cozza." In addition, the Examiner has argued that "Arnold teaches an alternative to decryption when searching for viruses in an encrypted file," and that "[a]s such, the ordinary person skilled in the art would have been motivated to implement the teachings of Arnold in place of the decompression and searching of Cozza." Further, the Examiner has argued that "Cozza does not teach away from the teachings of Arnold, but rather disclosed an alternative to the method of Arnold," and that "[n]owhere in Cozza is there any teaching that would cause one of ordinary skill in the art to ignore or avoid the teachings of Arnold."

Appellant respectfully disagrees. Cozza discloses that "[a] compressed file may have already been decompressed" but that "[i]f the file does require decompression, then it is decompressed" (Col. 7, lines 19-21 – emphasis added). In fact, Cozza only teaches that after such decompression, the file is scanned (see steps 84, 86, 90 and 94 in Figure 4D of Cozza). Thus, Cozza expressly discloses that a compressed file requires decompression prior to being scanned.

Arnold, on the other hand, relates to "detect[ing] and locat[ing] patterns that are present within data that has been encrypted" (see Abstract – emphasis added). As even noted by the Examiner, "Arnold teaches an alternative to decryption when searching for viruses in an encrypted file." Clearly, there would have been no motivation or suggestion to combine Arnold, which only relates to searching for patterns within encrypted data, with the teachings of Cozza, which only scans compressed data after such data is decompressed. Thus, the Examiner's proposed combination is inappropriate. Similarly, searching for patterns within encrypted data, as in Arnold, does in fact *teach away* from only scanning compressed data after such data is decompressed, as in Cozza.

Thus, appellant respectfully asserts that the first element of the *prima facie* case of obviousness has not been met, as noted above. More importantly, appellant also respectfully asserts that the

third element of the *prima facie* case of obviousness has not been met by the prior art reference excerpts relied on by the Examiner. For example, with respect to the independent claims, the Examiner has relied on Col. 2, lines 54-65 and Col. 6, lines 6-45 from the Cozza reference to make a prior art showing of appellant's claimed "reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file" (see this or similar, but not necessarily identical language in the independent claims).

Appellant respectfully asserts that the excerpts from Cozza relied on by the Examiner merely teach that "the file's cache information is checked to see if it is marked as having been previously infected by some virus which changes a file's resource fork size" (Col. 6, lines 21-23 – emphasis added) and that "[i]f a file's cache information is not marked as having been previously infected... then the file's current resource fork size is compared with the resource fork size stored in the file's cache information... to see if they are within some predetermined tolerance" (Col. 6, lines 29-34 – emphasis added).

Further, the excerpts teach that "the resource fork... may contain a kind of small database which is used to contain many kinds of data, including application code, icons, preferences, strings, templates, and other such items" (Col. 2, lines 55-60 – emphasis added). Also, the excerpts disclose that "[i]f the compressed file sizes... are different or if there are some viruses that could infect this file without changing its compressed size... then fork size information for this file is obtained," which "could involve decompressing the file, opening the file, or executing some special system or other code in order to obtain this information" (Col. 6, lines 14-20 – emphasis added).

However, merely disclosing comparing the resource fork size with a cached resource fork size, where the resource fork may contain a small database including application code, icons, preferences, strings, and templates, as in Cozza, fails to teach "reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program" (emphasis added), in the context claimed by appellant. Further, merely teaching that fork size information for a file may be obtained by decompressing the file, opening

the file, or executing some special system, as in Cozza, does not teach "reading resource data within said packed computer file, said resource data... readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file" (emphasis added), in the context claimed by appellant.

Further still, the Examiner has argued that Cozza teaches that "the compressed file is not decompressed in order to read the resource forks information." Appellant respectfully disagrees and notes that Cozza teaches that "fork size information for this file is obtained" and that "[t]his could involve decompressing the file" (Col. 6, lines 17-19 – emphasis added).

In the Office Action mailed 09/17/2007, the Examiner has argued that "the resource fork size falls within the scope of resource data," and that "Cozza disclosed that the resource fork may include application code, icons, preferences, strings, templates, and other such items" which "all fall under resource data as well." The Examiner has further argued that "by reading the resource fork size, which is part of the resource data, Cozza has met the limitation of [appellant's] claim." Still yet, the Examiner has argued that "Cozza specifically says that this may involve decompression, **or** executing some other special system or other code in order to obtain this information," such that "Cozza disclosed that decompression was not required."

Appellant respectfully disagrees. Only reading a resource fork size, as in Cozza, does not meet appellant's claimed "reading resource data within said packed computer file, **said** resource data [that is read] **specifying program resource items** used by said known computer program" (emphasis added), in the context claimed by appellant. Appellant emphasizes that merely reading a **size** of a resource fork, where the resource fork may include application code, icons, preferences, strings, templates, and other such item, as noted by the Examiner, does not include reading the resource fork itself, as the Examiner seems to suggest, and especially does not meet appellant's specifically claimed "said resource data [that is read] specifying program resource items used by said known computer program" (emphasis added), as claimed.

Additionally, merely alleging that Cozza discloses using decompression, or executing some other special system or other code in order to obtain this information, as noted by the Examiner, fails to even suggest "reading resource data within said packed computer file, said resource data...

readable by a computer operating system without dependence upon which unpacking algorithm **is used** by said packed computer file" (emphasis added), in the context claimed by appellant.

Further, the Examiner has relied on Figure 5 in Cozza to make a prior art showing of appellant's claimed technique "wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data" (see this or similar, but not necessarily identical language in the independent claims).

Appellant respectfully asserts that the only "flag" shown in Figure 5 of Cozza is simply utilized to indicate the presence of a virus. In addition, appellant points out that the flag is included in a scan information cache file which "includes data that has been accumulated during the scanning of files" (Col. 5, lines 17-21). Clearly, a flag that indicates whether a virus is present in scanned data, as in Cozza, fails to teach that "generated fingerprint data includes a flag indicating which data is included **within said generated fingerprint data**" (emphasis added), as claimed.

Moreover, the Examiner has argued that "[i]t further would have been obvious that because the fingerprint data represented the file during comparison, and the flags of Cozza indicated the viruses found in the file, the fingerprint data would have included a flag indicating which data (viruses) was included within said fingerprint data."

Appellant respectfully disagrees and again asserts that appellant claims that "said resource data of said packed computer file is processed to generate fingerprint data" where "said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data," when read in context. Clearly, the fingerprint claimed by appellant is generated from processed resource data, in the context claimed, and is not merely representative of the file during comparison, as alleged by the Examiner. Furthermore, appellant respectfully asserts that the flags disclosed in Cozza are only included in the scan information cache file which "includes data that has been accumulated during the scanning of files" (see Col. 5, lines 17-19 in Cozza), and not in the file itself in which a virus was found. To this end, even a hash of the file that includes the viruses would not itself include a flag indicating any sort of viruses, as argued by the Examiner. Thus, it would not have been obvious for "said generated fingerprint

data [to include] a flag indicating which data is included within said generated fingerprint data," in the context claimed.

In the Office Action mailed 09/17/2007, the Examiner has argued that "the claim language does not require that the flag indicates all of the data that was included in the generated fingerprint data," and that "[e]ach set of flags of Cozza indicates which viruses are contained within a specific file." The Examiner has further argued that "the file data is used to generate the hash portion of the fingerprint," and that "[t]he flags indicate which viruses, and thus which data, was included in the file data which is hashed, which falls within the scope of the claim language."

Appellant respectfully disagrees. Cozza only discloses that the flag is included in a scan information cache file that "includes data that has been accumulated during the scanning of files" (Col. 5, lines 17-21). Clearly, a flag included in such a scan information cache file, as in Cozza, does not meet appellant's claimed technique "wherein said **generated fingerprint data** includes a flag indicating which data is included within said generated fingerprint data" (emphasis added), particularly when "said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file," in the context claimed.

Still yet, with respect to the independent claims, the Examiner has relied Col. 6, lines 29-45 from the Cozza reference (reproduced below) to make a prior art showing of appellant's claimed technique "wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size" (see this or similar, but not necessarily identical language in the independent claims).

> "If a file's cache information is not marked as having been previously infected by some virus which changes a file's resource fork size, then **the file's current resource fork size is compared with the resource fork size stored in the file's cache information in step 66 to see if they are within some predetermined tolerance**. The tolerance in this step is determined based upon the size of viruses infecting a file's resource fork on the Apple Macintosh computer, upon the type of file being infected, and upon the typical size changes that might occur in Macintosh applications and other executable files due to minor changes by which the file might modify itself. This tolerance may vary from one file to another depending on file type and other factors. If these sizes are not within the predetermined tolerance, then flags are set for all viruses that might cause this file's resource fork to change

size when infecting it in step 68." (Cozza, Col. 6, lines 29-45 - emphasis added)

Appellant respectfully asserts that the excerpt from Cozza relied upon by the Examiner merely discloses comparing "the file's current resource fork size... with the resource fork size stored in the file's cache information in step 66 to see if they are within some predetermined tolerance" (emphasis added). Clearly, simply disclosing checking a current or stored resource fork size, as in Cozza, fails to even suggest a technique "wherein said generated fingerprint data includes a **location** within said resource data of said packed computer file of an entry specifying a program resource item having a largest size" (emphasis added), as claimed by appellant.

In the Office Action mailed 09/17/2007, the Examiner has argued that "[b]ecause the file contains the resource fork and resource items, and the hash is taken of the file, the signature includes a number of resource items specified within the resource fork, including the location of the entries of the resource items, including the largest resource item." In addition, the Examiner has argued that "[s]imilarly, because the hash is of the file, the hash includes all locations within the file."

Appellant respectfully disagrees. Simply because a file allegedly contains the resource fork, as noted by the Examiner, does not inherently suggest that "said generated fingerprint data includes a **location** within said resource data of said packed computer file **of an entry** specifying a program resource item having a largest size" (emphasis added), as specifically claimed by appellant.

Moreover, with respect to the independent claims, the Examiner has relied on Figure 5 from Cozza and Page 21, "PE File Base Relocations" in Pietrek to make a prior art showing of appellant's claimed technique "wherein said generated fingerprint data includes a checksum value calculated in dependence upon: a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file" (see this or similar, but not necessarily identical language in the independent claims).

Appellant respectfully asserts that the figure from Cozza relied upon by the Examiner merely discloses "cache data structures" where a "new cache includes data that has been accumulated

during the scanning of files, data about the cache itself, i.e. its version, volume creation date, file id, and checksum, and scan information for each file scanned" (Col. 5, lines 17-21 - emphasis added). Additionally, appellant notes that the excerpt from Pietrek relied on by the Examiner merely teaches that "[w]hen the linker creates an EXE file, it makes an assumption about where the file will be mapped into memory" and "[b]ased on this, the linker puts the real addresses of code and data items into the executable file" (Page 21, "PE File Base Relocations," paragraph 1).

However, merely disclosing a cache data structure which includes a cache checksum, in addition to teaching that a linker puts code and data into an executable file, fails to even *suggest* a technique "wherein said generated fingerprint data includes a checksum value calculated in dependence upon: a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file" (emphasis added), as claimed by appellant.

Additionally, the Examiner has argued that it "would have been obvious that the checksum of the file in this combination would have been dependent upon a number of said program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file... as Pietrek teaches that Win32 PE files are arranged in such a manner, as seen [in] Pietrek Fig. 5 and Table 13."

Appellant respectfully disagrees and points out that Figure 5 and Table 13 from Pietrek merely show a resource directory hierarchy example and the resources hierarchy for CLOCK.EXE. However, the table and figure relied upon by the Examiner in no way suggest "**fingerprint data** [that] includes a checksum value calculated in dependence upon: a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file" (emphasis added), as claimed by appellant.

In the Office Action mailed 09/27/2007, the Examiner has relied upon "the 'hash', as taught by Arnold, in the combination, as meeting the limitation of the checksum, and not the checksum of the cache disclosed by Cozza."

Appellant respectfully disagrees and asserts that that even the combination with Arnold does not meet appellant's specific claim language. For example, Arnold's mere disclosure of using a hashing function (Col. 10, lines 48-53), in addition to Pietrek's general disclosure of a resource directory hierarchy does not specifically teach a technique "wherein said generated fingerprint data includes a checksum value calculated in dependence upon: a number of program resource items **specified beneath each node** within hierarchically arranged resource data of said packed computer file" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #2: Claims 14, 30, 46, 62, 78, 94 and 98

With respect to dependent Claim 14 et al. and dependent Claim 98, the Examiner has admitted that Cozza, Arnold, and Pietrek do not disclose appellant's specific claim language, and thus has simply dismissed the same under Official Notice. Specifically, the Examiner has argued that "SHA, which shifts 1 bit to the left after each operation, was a well known checksum in the art at the time of the invention, and as such it would have been obvious to the ordinary person skilled in the art to have used SHA as the checksum."

Appellant respectfully disagrees and asserts that even assuming *arguendo* that it would have been obvious to shift 1 bit after each operation in calculating a checksum, such still does not address appellant's specific claim language, namely that "said checksum value is rotated between each item being added into said checksum" (see Claim 14 et al. - emphasis added) and that "said checksum value is rotated 1 bit to the left" (see Claim 98), as claimed. Thus, appellant's claimed technique simply would not have been obvious.

Appellant thus formally requests a specific showing of the subject matter in ALL of the claims in any future action. Note excerpt from MPEP below.

"If the [appellant] traverses such an [Official Notice] assertion the examiner should cite a reference in support of his or her position." See MPEP 2144.03.

In the Office Action mailed 09/17/2007, the Examiner has argued that Pages 442-445, and especially Fig. 18.7, of Schneier (Applied Cryptography, Second Edition) "shows the 'rotation' in SHA, called a left circular shift, and each block of data reads on the 'item'."

Appellant respectfully disagrees and asserts that the Examiner has relied upon the Schneier reference, which constitutes a reference separate from those in the relevant rejection under 35 U.S.C. 103(a). Further, it is noted that the Examiner has failed to cite specific motivation in the relevant reference(s) to support the case for combining the Schneier reference. The Examiner is reminded that the Federal Circuit requires that there must be some logical reason apparent from the evidence of record that would justify the combination or modification of references. In re Regel, 188 USPQ 132 (CCPA 1975). Thus, the reliance on the Schneier reference, on its face, is clearly improper.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

Group #3: Claims 15, 31, 47, 63, 79 and 95

With respect to dependent Claim 15 et al., the Examiner has relied upon Page 21, PE File Base Relocations, of the Pietrek reference to make a prior art showing of appellant's claimed technique "wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program."

Appellant respectfully asserts that the excerpt from Pietrek relied upon by the Examiner merely discloses that "[w]hen the linker creates an EXE file, it makes an assumption about where the file will be mapped into memory" and "[b]ased on this, the linker puts the real addresses of code and data items into the executable file" (Page 21). Further, the excerpt teaches that "[i]f... the

executable ends up being loaded somewhere else in virtual space" then "[t]he information stored in the .reloc section allows the PE loader to fix these addresses" (Page 21). However, the mere disclosure of a PE loader fixing the real addresses of code and data items when the executable is loaded somewhere else in virtual space, as in Pietrek, simply fails to even suggest "decompress[ing] said known computer program," much less a technique "wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program" (emphasis added), as claimed by appellant.

Again, appellant respectfully asserts that at least the first and third elements of the *prima facie* case of obviousness have not been met, since it would be *unobvious* to combine the references, as noted above, and the prior art excerpts, as relied upon by the Examiner, fail to teach or suggest all of the claim limitations, as noted above.

In view of the remarks set forth hereinabove, all of the independent claims are deemed allowable, along with any claims depending therefrom.

VIII CLAIMS APPENDIX (37 C.F.R. § 41.37(c)(1)(viii))

The text of the claims involved in the appeal (along with associated status information) is set forth below:

1. (Previously Presented) A computer program product in a computer storage medium comprising a computer program operable to control a computer to detect a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said computer program comprising:

resource data reading logic for reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

resource data comparing logic for generating characteristics of said resource data and for comparing said characteristics of said resource data with characteristics of known computer program resource data and for detecting a match with said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of said program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with said program resource items within said resource data of said packed computer file; and

sizes of said program resource items within said resource data of said packed computer file.

2. (Original) A computer program product as claimed in claim 1, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

3. (Previously Presented) A computer program product as claimed in claim 1, wherein said resource data comparing logic is operable to compare said resource data of said packed computer file with characteristics of a plurality of known computer programs to detect if said packed computer file contains one of said plurality of known computer programs.

4. (Cancelled)

5. (Original) A computer program product as claimed in claim 1, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;

bitmap data;

menu data; and

language data.

6. (Previously Presented) A computer program product as claimed in claim 1, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

7. (Original) A computer program product as claimed in claim 6, wherein said storage location of said resource item is specified as an relative offset value.

8. (Previously Presented) A computer program product as claimed in claim 1, wherein said resource data of said packed computer file specifies for each resource item a size of said resource item.

9.-11. (Cancelled)

12. (Previously Presented) A computer program product as claimed in claim 1, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

13. (Cancelled)

14. (Previously Presented) A computer program product as claimed in claim 1, wherein said checksum value is rotated between each item being added into said checksum.

15. (Original) A computer program product as claimed in claim 1, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

16. (Original) A computer program product as claimed in claim 1, wherein said packed computer file is a Win32 PE file.

17. (Previously Presented) A computer program product in a computer storage medium comprising a computer program operable to control a computer to generate data for detecting a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said computer program comprising:

resource data reading logic for reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

characteristic data generating logic for generating characteristic data associated with said resource data for comparison with characteristic data of known computer program resource data

to detect a match with said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with program resource items within said resource data of said packed computer file; and

sizes of program resource items within said resource data of said packed computer file.

18. (Original) A computer program product as claimed in claim 17, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

19. (Previously Presented) A computer program product as claimed in claim 17, wherein said characteristic data generating logic is operable to generate characteristic data from a plurality of known computer programs to enable detection of any of said plurality of known computer programs within said packed computer file.

20. (Cancelled)

21. (Original) A computer program product as claimed in claim 17, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;

bitmap data;

menu data; and

language data.

22. (Previously Presented) A computer program product as claimed in claim 17, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

23. (Original) A computer program product as claimed in claim 22, wherein said storage location of said resource item is specified as an relative offset value.

24. (Previously Presented) A computer program product as claimed in claim 17, wherein said resource data of said packed computer file specifies for each resource item a size of said resource item.

25.-27. (Cancelled)

28. (Previously Presented) A computer program product as claimed in claim 17, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

29. (Cancelled)

30. (Previously Presented) A computer program product as claimed in claim 17, wherein said checksum value is rotated between each item being added into said checksum.

31. (Original) A computer program product as claimed in claim 17, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

32. (Original) A computer program product as claimed in claim 17, wherein said packed computer file is a Win32 PE file.

33. (Previously Presented) A method of controlling a computer to detect a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said method comprising the steps of:

reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

generating characteristics of said resource data and comparing said characteristics of said resource data with characteristics of known computer program resource data and detecting a match with characteristics of said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with program resource items within said resource data of said packed computer file; and

sizes of program resource items within said resource data of said packed computer file.

34. (Original) A method as claimed in claim 33, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

35. (Previously Presented) A method as claimed in claim 33, wherein said step of comparing compares said resource data of said packed computer file with characteristics of a plurality of known computer programs to detect if said packed computer file contains one of said plurality of known computer programs.

36. (Cancelled)

37. (Original) A method as claimed in claim 33, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;

bitmap data;

menu data; and

language data.

38. (Previously Presented) A method as claimed in claim 33, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

39. (Original) A method as claimed in claim 38, wherein said storage location of said resource item is specified as an relative offset value.

40. (Previously Presented) A method as claimed in claim 33, wherein said resource data of said packed computer file specifies for each resource item a size of said resource item.

41.-43. (Cancelled)

44. (Previously Presented) A method as claimed in claim 33, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

45. (Cancelled)

46. (Previously Presented) A method as claimed in claim 33, wherein said checksum value is rotated between each item being added into said checksum.

47. (Original) A method as claimed in claim 33, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

48. (Original) A method as claimed in claim 33, wherein said packed computer file is a Win32 PE file.

49. (Previously Presented) A method of controlling a computer to generate data for detecting a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said method comprising the steps of:

reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

generating characteristic data associated with said resource data for comparison with characteristic data of known computer program resource data and detecting a match with said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with program resource items within said resource data of said packed computer file; and

sizes of program resource items within said resource data of said packed computer file.

50. (Original) A method as claimed in claim 49, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

51. (Previously Presented) A method as claimed in claim 49, wherein said step of generating generates characteristic data from a plurality of known computer programs to enable detection of any of said plurality of known computer programs within said packed computer file.

52. (Cancelled)

53. (Original) A method as claimed in claim 49, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;
bitmap data;
menu data; and
language data.

54. (Previously Presented) A method as claimed in claim 49, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

55. (Original) A method as claimed in claim 54, wherein said storage location of said resource item is specified as an relative offset value.

56. (Previously Presented) A method as claimed in claim 49, wherein said resource data of said packed computer file specifies for each resource item a size of said resource item.

57.-59. (Cancelled)

60. (Previously Presented) A method as claimed in claim 49, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

61. (Cancelled)

62. (Previously Presented) A method as claimed in claim 49, wherein said checksum value is rotated between each item being added into said checksum.

63. (Original) A method as claimed in claim 49, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

64. (Original) A method as claimed in claim 49, wherein said packed computer file is a Win32 PE file.

65. (Previously Presented) Apparatus for detecting a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said apparatus comprising:

a resource data reader for reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

a resource data comparator for generating characteristics of said resource data and for comparing said characteristics of said resource data with characteristics of known computer program resource data for detecting a match with said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with program resource items within said resource data of said packed computer file; and

sizes of program resource items within said resource data of said packed computer file.

66. (Original) Apparatus as claimed in claim 65, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

67. (Previously Presented) Apparatus as claimed in claim 65, wherein said resource data comparator is operable to compare said resource data of said packed computer file with characteristics of a plurality of known computer programs to detect if said packed computer file contains one of said plurality of known computer programs.

68. (Cancelled)

69. (Original) Apparatus as claimed in claim 65, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;

bitmap data;

menu data; and

language data.

70. (Previously Presented) Apparatus as claimed in claim 65, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

71. (Original) Apparatus as claimed in claim 70, wherein said storage location of said resource item is specified as an relative offset value.

72. (Previously Presented) Apparatus as claimed in claim 65, wherein said resource data of said packed computer file specifies for each resource item a size of said resource item.

73.-75. (Cancelled)

76. (Previously Presented) Apparatus as claimed in claim 65, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

77. (Cancelled)

78. (Previously Presented) Apparatus as claimed in claim 65, wherein said checksum value is rotated between each item being added into said checksum.

79. (Original) Apparatus as claimed in claim 65, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

80. (Original) Apparatus as claimed in claim 65, wherein said packed computer file is a Win32 PE file.

81. (Previously Presented) Apparatus for generating data for detecting a known computer program within a packed computer file, said packed computer file being unpacked upon execution, said apparatus comprising:

a resource data reader for reading resource data within said packed computer file, said resource data specifying program resource items used by said known computer program and readable by a computer operating system without dependence upon which unpacking algorithm is used by said packed computer file; and

a characteristic data generator for generating characteristic data associated with said resource data for comparison with characteristic data of known computer program resource data and for detecting a match with said known computer program indicative of said packed computer file containing said known computer program;

wherein said resource data of said packed computer file is processed to generate fingerprint data and to compare said generated fingerprint data with known computer program fingerprint data;

wherein said generated fingerprint data includes a number of program resource items specified within said resource data of said packed computer file;

wherein said generated fingerprint data includes a flag indicating which data is included within said generated fingerprint data;

wherein said generated fingerprint data includes a location within said resource data of said packed computer file of an entry specifying a program resource item having a largest size;

wherein said generated fingerprint data includes a checksum value calculated in dependence upon:

a number of program resource items specified beneath each node within hierarchically arranged resource data of said packed computer file;

string names associated with program resource items within said resource data of said packed computer file; and

sizes of program resource items within said resource data of said packed computer file.

82. (Original) Apparatus as claimed in claim 81, wherein said known computer program is one of:

a Trojan computer program; and

a worm computer program.

83. (Previously Presented) Apparatus as claimed in claim 81, wherein said characteristic data generator is operable to generate characteristic data from a plurality of known computer programs to enable detection of any of said plurality of known computer programs within said packed computer file.

84. (Cancelled)

85. (Original) Apparatus as claimed in claim 81, wherein said program resource items used by said known computer program include one or more of:

icon data;

string data;

dialog data;

bitmap data;

menu data; and

language data.

86. (Previously Presented) Apparatus as claimed in claim 81, wherein said resource data of said packed computer file specifies for each resource item a storage location of said resource item.

87. (Original) Apparatus as claimed in claim 86, wherein said storage location of said resource item is specified as an relative offset value.

88. (Previously Presented) Apparatus as claimed in claim 81, wherein said resource for each resource item a size of said resource item. data of said packed computer file specifies

89.-91. (Cancelled)

92. (Previously Presented) Apparatus as claimed in claim 81, wherein said generated fingerprint data includes timestamp data indicative of a time of compilation of said known computer program.

93. (Cancelled)

94. (Previously Presented) Apparatus as claimed in claim 81, wherein said checksum value is rotated between each item being added into said checksum.

95. (Original) Apparatus as claimed in claim 81, wherein said packed computer file includes an unpacking computer program which upon execution decompresses said known computer program.

96. (Original) Apparatus as claimed in claim 81, wherein said packed computer file is a Win32 PE file.

97. (Cancelled)

98. (Previously Presented) A computer program product as claimed in claim 14, wherein said checksum value is rotated 1 bit to the left.

IX EVIDENCE APPENDIX (37 C.F.R. § 41.37(c)(1)(ix))

There is no such evidence.

X RELATED PROCEEDING APPENDIX (37 C.F.R. § 41.37(c)(1)(x))

N/A

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 971-2573. For payment of any additional fees due in connection with the filing of this paper, the Commissioner is authorized to charge such fees to Deposit Account No. 50-1351 (Order No. NAI1P467).

Respectfully submitted,

By: /KEVINZILKA/ Date: March 4, 2008
Kevin J. Zilka
Reg. No. 41,429

Zilka-Kotab, P.C.
P.O. Box 721120
San Jose, California 95172-1120
Telephone: (408) 971-2573
Facsimile: (408) 971-4660